Data

The file `neonTickMetData.R` (written by John Foster) matched data from NEON tick surveys with daily meteorological products from NEON sensors. The files produce the CSV files we work with for this analysis. These files are read in below.

Data structure There are 2 data streams when you download the tick dataset. There is taxonomic data and field data.

How is sampling done? The data are counts of lone star ticks. Tick sampling is done by dragging or flagging. A person walks a transect and drags a one square meter cloth behind them. Afterwards they count the ticks. From the field data, they have information on numbers. For the taxonomic data, they ID the nymphs and adults down to species.

What's the life cycle? The nymph emerges in the spring and try to take a blood meal. We might see adults later in the year or next year if they are successful in feeding. They only take one blood meal per life stage.

Correction for effort These counts have been corrected for effort. The effort that went into sampling each row is assumed to be the same.

Data are daily totals The abundances are collated (summed) across the site. There is one meteorological station per site.

How big are the sites? Are all the NEON sites similar sizes, or is there substantial variation? How many different habitat types does each NEON site encompass? I have these questions because they might be useful for thinking about if we would expect different amounts of variability between sites.

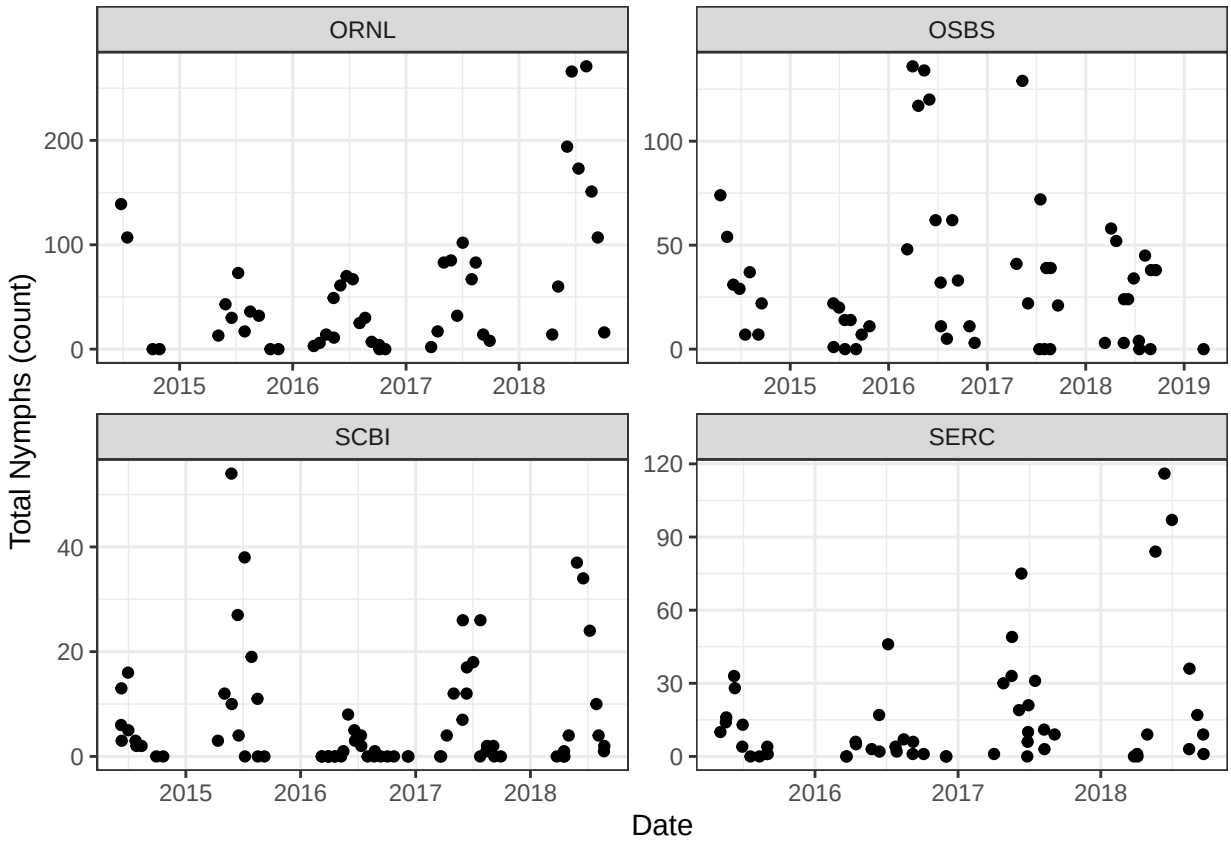
Climate covariates Total precipitation is in millimeters (mm). Relative humidity is a percentage. Temperature is in celsius. There is a lot of missing data (unknown reason, perhaps sensors were not online yet?), especially for precipitation.

Data exploration

Here we read in the data.

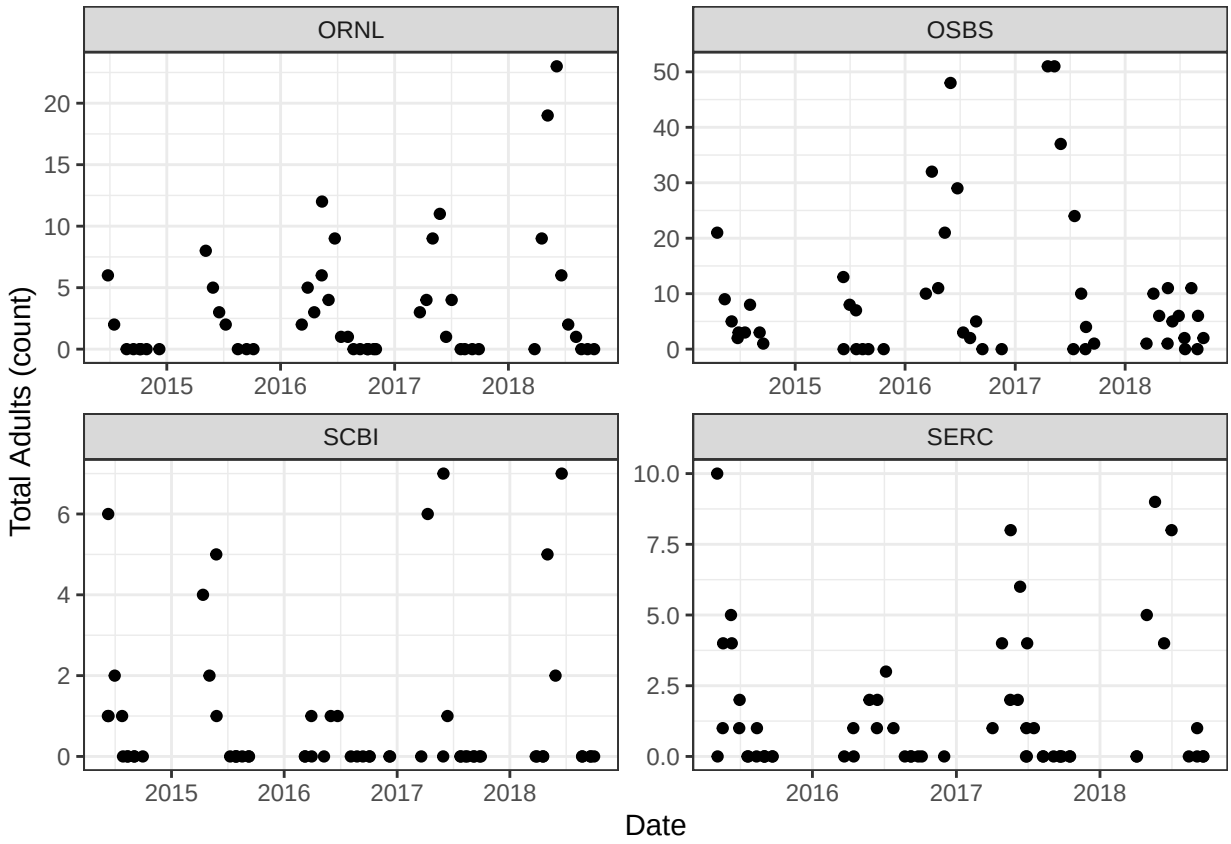
Let's start by looking at the data. Here's the nymph data to begin with:

```
nymphDataWithMet <- nymphDataWithMet %>%
  dplyr::mutate(day = as.Date(Day))

 ggplot(data=nymphDataWithMet) +
   geom_point(aes(x=day,y=totalNymph)) +
   facet_wrap(~Site,scale='free') +
   ylab("Total Nymphs (count)") +
   xlab("Date") +
   theme_bw()
```



And here's the adult data:

```
adultDataWithMet <- adultDataWithMet %>%
  dplyr::mutate(day = as.Date(Day))

 ggplot(data=adultDataWithMet) +
   geom_point(aes(x=day,y=totalAdult)) +
   facet_wrap(~Site,scale='free') +
   ylab("Total Adults (count)") +
   xlab("Date") +
   theme_bw()
```



Some first notes are that site `SERC` does not have data for 2014. So that's one less year than the other sites. Site `SCBI` seems to have the least within-year variation, and site `OSBS` has the most within-year variation.

What's the outstanding question?

John suggests focusing on the time component and forecasting the 2018 season.

Shannon suggests that we may want to focus on one site to start.

A starting place could be to begin with fitting a random walk to the data. We could then try to figure out which covariates to include in a dynamic linear model.

We could also build a forecast model at one site and see if it forecasts another site.

How will we collaborate over the week?

- Email
- Chat over the Slack (link to private thread)
- Github (link to Github repo).

State-space model

First goal: set up a random walk for one of the sites.

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
library(MCMCvis)
```

Work with only one site (`SERC`), and set all the validation data points to `NA`.

```
data = adultDataWithMet %>%
  dplyr::filter(Site=="SERC") %>%
  #dplyr::filter(trainValidate=="Train")
  dplyr::mutate(totalAdultOriginal = totalAdult,
                totalAdult = ifelse(trainValidate=="Train",totalAdult,NA))
```

Time and y are the two variables we're interested in.

```
time = as.Date(data$Day)
y = data$totalAdult
```

We'll start by trying to fit a random walk model. Change the data model to a Poisson. The linear predictor for the Poisson is with a log-link. The mean of a Poisson can be explained by a linear process.

One of the problems we're encountering is that we have a lot of 0s in our data, so log-transforming those values immediately turns them into `-Inf`. What are some ways of dealing with this for random walk models?

What I did was modify the process model to use a log-normal distribution instead of a normal distribution. I set the mean of the log normal (`mu` in the code) to have a lower bound of 0.0001, so that it can't take on the value of 0.

I'm interested in seeing how this compares to what others did or how others dealt with this!

```
RandomWalk = "
model{

  #### Priors
  x[1] ~ dlnorm(x_ic,tau_add[1])
  #tau_obs ~ dgamma(a_obs,r_obs)
  tau_add ~ dgamma(a_add,r_add)

  #### Process Model
  for(t in 2:n){
    # need to set lower bound on mu
    mu[t] = log(max(0.0001, x[t-1]))
    x[t]~dlnorm(mu[t],tau_add)
  }

  #### Data Model
  for(t in 1:n){
    y[t] ~ dpois(x[t])
  }

}
"
```

Define the data as a list.

```
data <- list(y=y,
             n=length(y),
```

```
            x_ic=log(y[1]),
            a_add=1,
            r_add=1)
```

Define initial conditions.

```
nchain = 3
init <- list()
for(i in 1:nchain){
  y.samp = sample(y,length(y),replace=TRUE)
  init[[i]] <- list(tau_add=1/var(y.samp))#,times=length(y)))
}
```

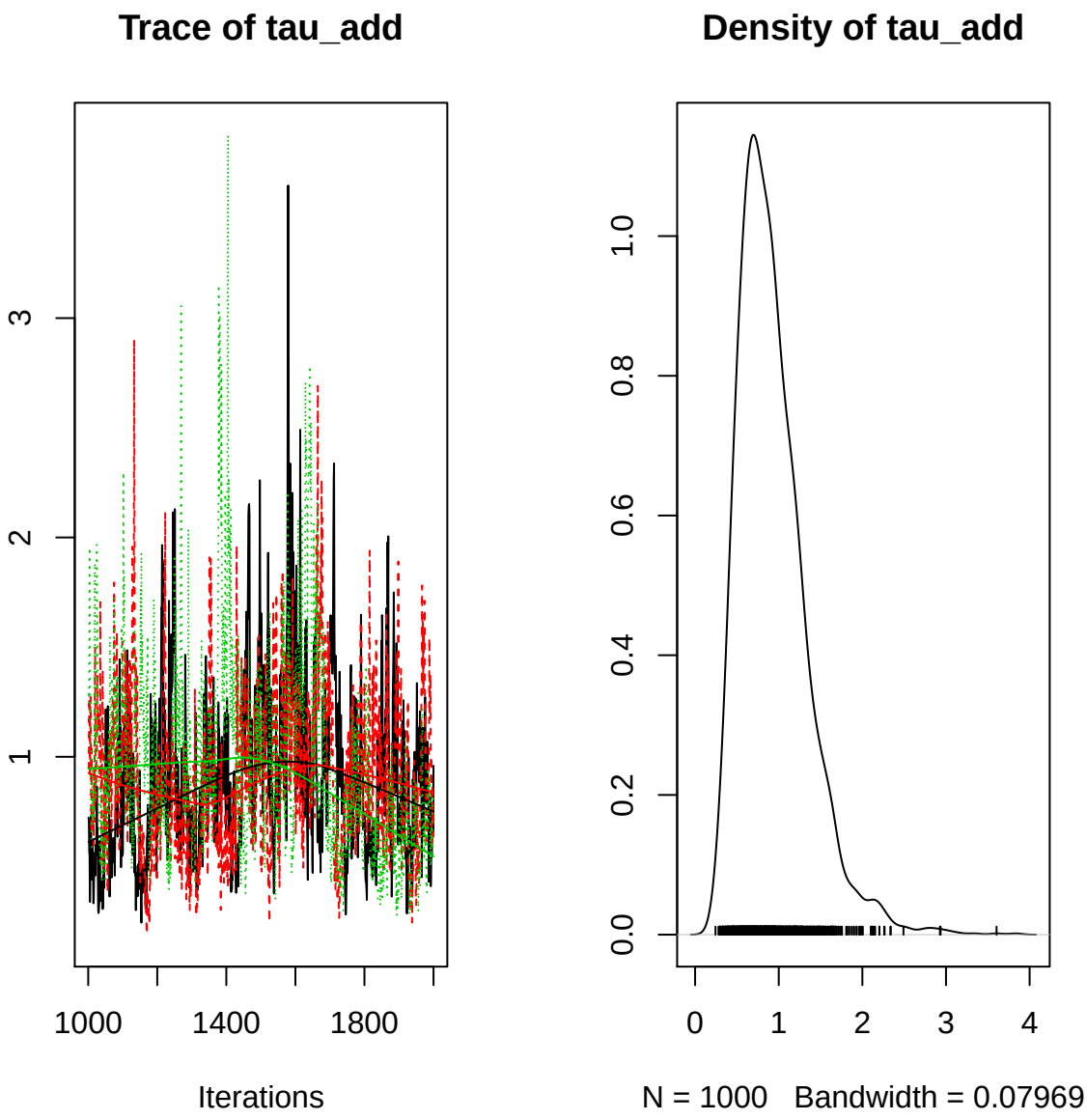
Send model, data, and inits to JAGS.

```
j.model   <- jags.model (file = textConnection(RandomWalk),
                            data = data,
                            inits = init,
                            n.chains = 3)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 53
##    Unobserved stochastic nodes: 70
##    Total graph size: 248
##
## Initializing model
```

Look at the precision parameter after burn-in.

```
## burn-in
jags.out   <- coda.samples (model = j.model,
                            variable.names = c("tau_add"),
                                n.iter = 1000)
plot(jags.out)
```



Run model, sampling the parameters in the model.

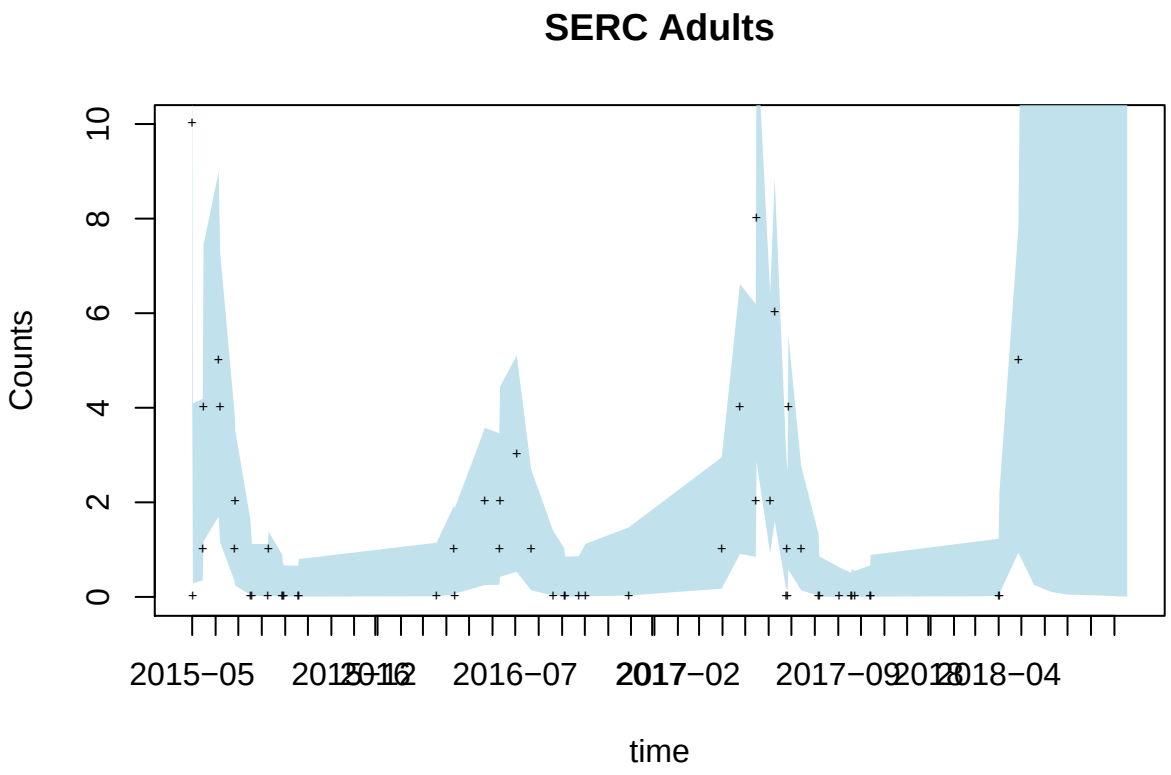
```
jags.out   <- coda.samples (model = j.model,
                            variable.names = c("x","mu","tau_add"),
                                n.iter = 10000)
```

Visualization

First, I plot the forecast using base R and the `ecoforecastR::ciEnvelope` function. Note there should be a change to the confidence intervals (line with `apply` because the model was not fit on a log scale).

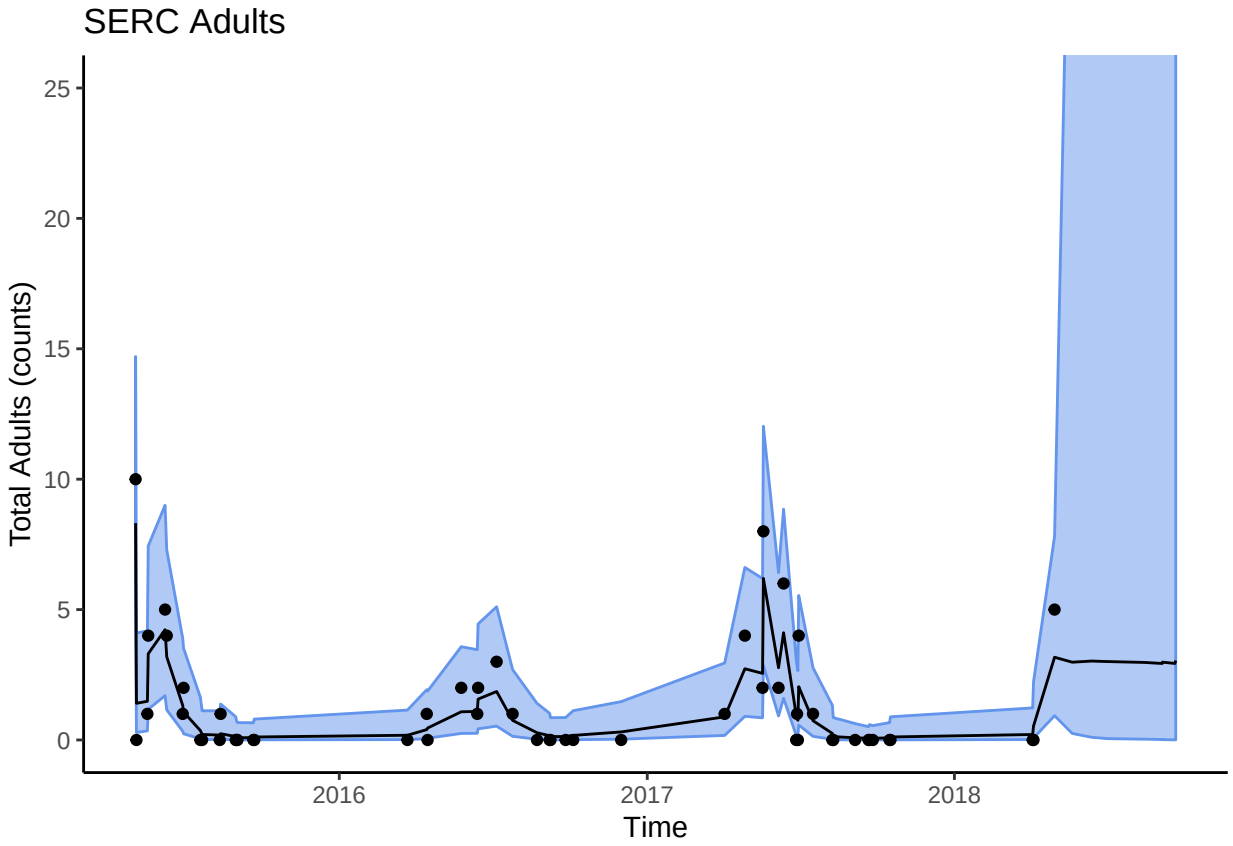
```
time.rng = c(1,length(time)) ## adjust to zoom in and out
out <- as.matrix(jags.out)
x.cols <- grep("^x",colnames(out)) ## grab all columns that start with the letter x
## remove exp() because model not on log scale
ci <- apply(out[,x.cols],2,quantile,c(0.025,0.5,0.975)) ## model was NOT fit on log scale
```

```
plot(time,ci[2,],type='n',ylim=c(0,range(y,na.rm=TRUE)[2]),
     ylab="Counts",xlim=time[time.rng],main="SERC Adults")
## adjust x-axis label to be monthly if zoomed
if(diff(time.rng) < 100){
  axis.Date(1, at=seq(time[time.rng[1]],time[time.rng[2]],by='month'), format = "%Y-%m")
}
ecoforecastR::ciEnvelope(time,ci[1,],ci[3,],col=ecoforecastR::col.alpha("lightBlue",0.75))
points(time,y,pch="+",cex=0.5)
```



I then made a similar plot with ggplot.

```
sum.data=data.frame(time,t(ci))

dataForPlotting = adultDataWithMet %>%
  dplyr::filter(Site=="SERC") %>%
  dplyr::filter(trainValidate=="Train")

ggplot() +
    geom_ribbon(data=sum.data,aes(x=time,ymin=X2.5.,ymax=X97.5.),
                color="cornflowerblue",fill="cornflowerblue",alpha=0.5) +
      geom_line(data=sum.data,aes(x=time,y=X50.),color="black") +
  geom_point(data=dataForPlotting,aes(x=day,y=totalAdult)) +
  theme_classic() +
  xlab("Time") + ylab("Total Adults (counts)") +  ggtitle("SERC Adults") +
  coord_cartesian(ylim = c(0, 25))
```



Forecasting

Now forecasting.

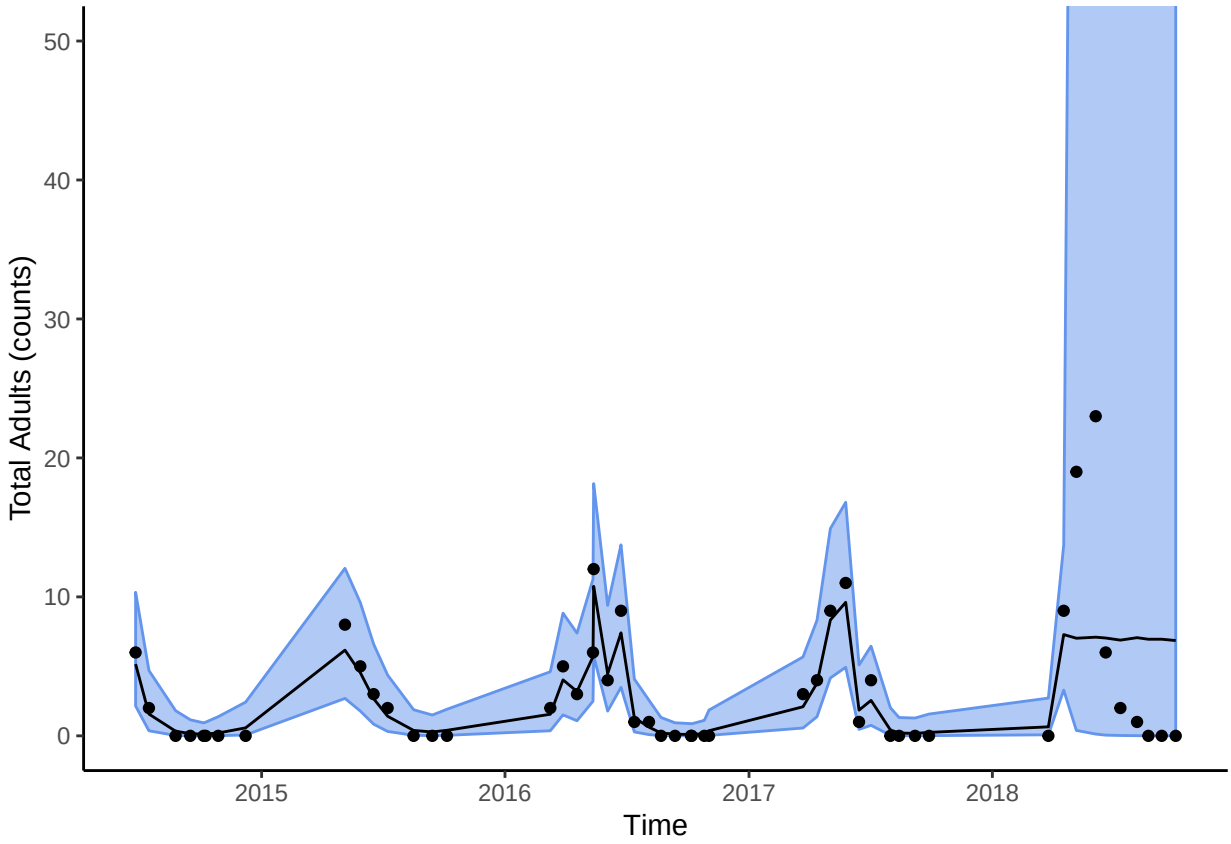
```
# define a function

fitRandomWalk <- function(data = adultDataWithMet, siteName = "SERC", jagsModel = RandomWalk,  forecast=

  tmp <- data %>%
    dplyr::filter(Site==siteName)

  if (forecast) {tmp$totalAdult[tmp$trainValidate=="Validate"] <- NA}
    else {tmp <- tmp[tmp$trainValidate=="Train",]}

  time = as.Date(tmp$Day)
  y = tmp$totalAdult

  dataJags <- list(y=y,
                   n=length(y),
                   x_ic=log(y[1]),
                   a_add=1,
                   r_add=1)

  nchain = 3
  init <- list()
  for(i in 1:nchain){
```

```
    y.samp = sample(y,length(y),replace=TRUE)
    init[[i]] <- list(tau_add=1/var(y.samp))
  }

  j.model <- jags.model (file = textConnection(jagsModel),
                         data = dataJags,
                         inits = init,
                         n.chains = 3)

  jags.out   <- coda.samples (model = j.model,
                              variable.names = c("x","mu","tau_add"),
                              n.iter = 10000)
}
```

```
jags.out <- fitRandomWalk(data = adultDataWithMet, siteName = "ORNL", jagsModel = RandomWalk, forecast="
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 42
##    Unobserved stochastic nodes: 59
##    Total graph size: 204
##
## Initializing model
```

```
out <- as.matrix(jags.out)
x.cols <- grep("^x",colnames(out)) ## grab all columns that start with the letter x
## remove exp() because model not on log scale
ci <- apply(out[,x.cols],2,quantile,c(0.025,0.5,0.975)) ## model was NOT fit on log scale

time = as.Date(adultDataWithMet[adultDataWithMet$Site=="ORNL",]$Day)

sum.data=data.frame(time,t(ci))

dataForPlotting = adultDataWithMet %>%
  dplyr::filter(Site=="ORNL") #%>%
 # dplyr::filter(trainValidate=="Train")

ggplot() +
  geom_ribbon(data=sum.data,aes(x=time,ymin=X2.5.,ymax=X97.5.),
              color="cornflowerblue",fill="cornflowerblue",alpha=0.5) +
  geom_line(data=sum.data,aes(x=time,y=X50.),color="black") +
  geom_point(data=dataForPlotting,aes(x=day,y=totalAdult)) +
  theme_classic() +
  xlab("Time") + ylab("Total Adults (counts)") +
  coord_cartesian(ylim = c(0, 50))
```



Note that this model misses all 0s.

Second goal: could then set up a workflow to run the other sites with the same model